Disentangle, Align and Generalize: Learning A Timing Predictor from Different Technology Nodes

Xinyun Zhang*, Binwu Zhu*, Fangzhou Liu, Ziyi Wang, Peng Xu, Hong Xu, Bei Yu

The Chinese University of Hong Kong

THE CHIPS TO SYSTEMS CONFERENCE

SHAPING THE NEXT GENERATION OF ELECTRONICS

Introduction

We propose a transfer learning framework that leverages extensive data at the preceding node and limited data at the target node to enhance the performance at the target node.

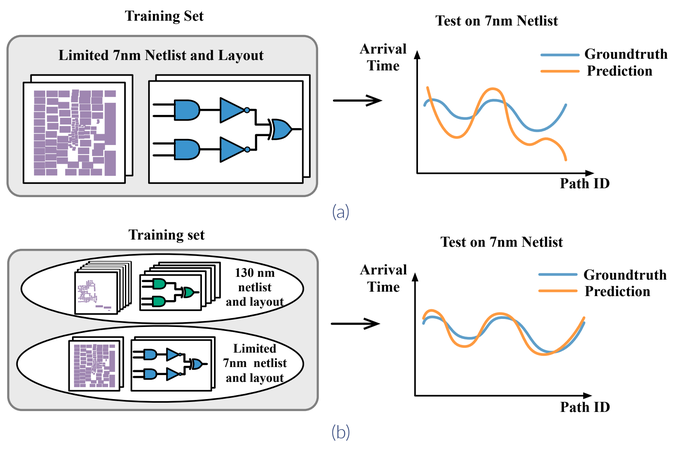


Figure 1. (a) Trained on limited 7nm netlist data; (b) Trained on both limited 7nm netlist data and 130nm netlist data.

Challenges:

- Netlist data consists of two kinds of knowledge: node-dependent (standard cell information) and design-dependent (functionalities). These two kinds of knowledge are highly intertwined in the netlist graph, making it difficult to leverage the common and transferable parts across different nodes.
- The arrival time values of different timing paths can vary dramatically, even by one or two orders of magnitude, which poses significant challenges for the ML-based regression model.
- The limited target node data makes the timing predictor susceptible to overfitting the training designs, which hinders the broad application of the learned model

Framework Overview

Our framework consists of three parts:

- Timing Path Feature Extractor
- Timing Feature Disentanglement and Alignment
- Bayesian-based Timing Prediction

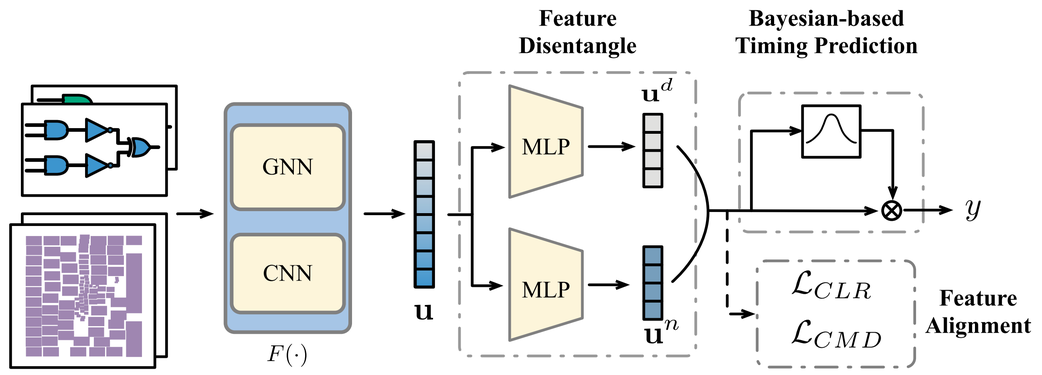


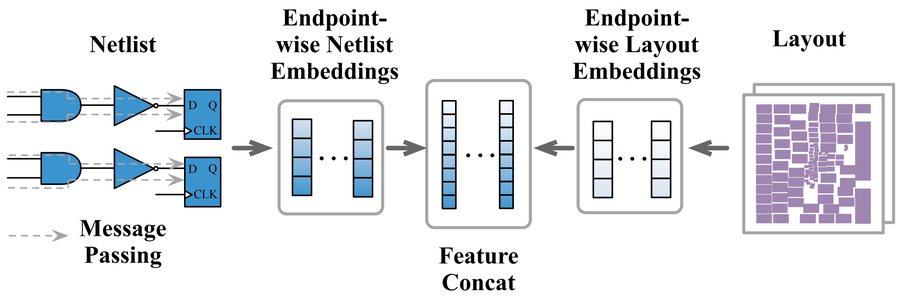
Figure 2. Overview of the method. Timing path feature alignment is only computed during the training stage.

Multimodal Timing Path Feature Extractor

- We first collect two types of input: the netlist graph $\mathcal{H}$ and the layout image set $\mathcal{X}$.
- We construct $\mathcal{H}$ as a heterogeneous graph with two types of edges: the net edge connecting a net's drive pin and one of its sink pins, and the cell edge connecting one of a cell's input pins and its output pin.
- Then, we use a GNN to propagate on $\mathcal{H}$ from the primary inputs to the endpoints to obtain the feature for each timing path $\mathcal{G}'$.
- Meanwhile, we mask each timing path with the pin locations on the layout image and use a convolution neural network (CNN) to extract the features of the timing path there.

The final feature can be denoted as:

$$\boldsymbol{u} = F(\mathcal{G}') = [\mathrm{GNN}(\mathcal{H}), \mathrm{CNN}(\mathcal{X})] \in \mathbb{R}^m. \tag{1}$$



Timing Feature Disentanglement and Alignment

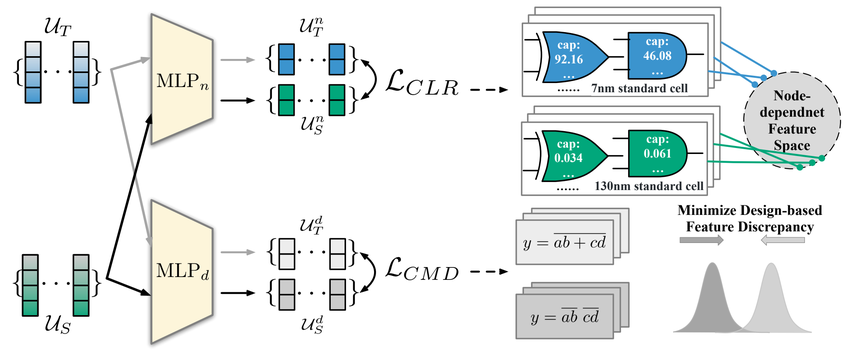


Figure 4. Timing path feature alignment.

We further adopt two loss functions as the objectives to disentangle the different features:

Node-based contrastive loss: Pull together the features from the same node and push apart the features from different nodes.

Design-based discrepancy loss: Close the gap between design-based features from different technology nodes.

Bayesian-based Timing Prediction

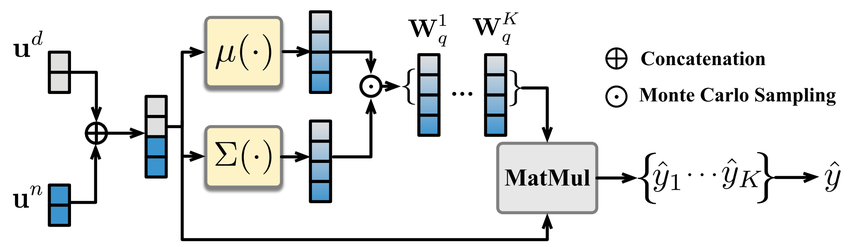


Figure 5. Timing path feature alignment.

- Modeling the final fully-connected layer as a distribution.
- Optimizing the evidence lower bound (ELBO).
- For the target technology node, using the node-based feature from the target node and the design-based feature from both nodes as the prior.

Experiments

Dataset Statistics:

Table 1. Statistics of the dataset (edp stands for endpoint, e_n and e_c denote net edge and cell edge, respectively)

Benchmark		Input information tech node	#pin	#edp	$\#e_n$	$\#e_c$
train	smallboom	$7nm$	694441	61764	488052	423344
	jpeg	$130nm$	1527166	39783	1150173	749294
	linkruncca	$130nm$	186546	17796	151617	84393
	spiMaster	$130nm$	99507	4739	75718	44917
	usbf_device	$130nm$	48104	4777	37557	21706
test	arm9	$7nm$	44469	2500	33065	29287
	chacha	$7nm$	35687	1986	25117	23083
	hwacha	$7nm$	1357798	61313	985057	922085
	or1200	$7nm$	1165114	172401	844443	658961
	sha3	$7nm$	794720	60323	552021	485596
Avg	train	$7nm\&130nm$	511153	25772	380623	264731
	test	$7nm$	679558	59705	487941	423802

Main results:

Table 2. The evaluation results on $7nm$ netlist data.

Baseline	DAC23 [3]-AdvOnly		DAC23 [3]-SimpleMerge		DAC23 [3]-ParamShare [5]		DAC23 [3]-PT-FT [1]		Ours	
	R^2 score	runtime	R^2 score	runtime	R^2 score	runtime	R^2 score	runtime	R^2 score	runtime
arm9	0.603	2.546	-2.069	2.546	0.567	2.546	0.837	2.546	**0.864**	2.621
chacha	0.624	1.188	-1.983	1.188	0.568	1.188	0.726	1.188	**0.890**	1.234
hwacha	0.170	5.229	-2.203	5.229	0.499	5.229	0.818	5.229	**0.828**	5.400
or1200	0.156	14.257	-6.037	14.257	0.240	14.257	0.209	14.257	**0.682**	14.793
sha3	0.425	1.690	-4.741	1.690	0.195	14.257	0.284	1.690	**0.785**	1.725
average	0.396	4.982	-3.407	4.982	0.414	4.982	0.575	4.982	**0.810**	5.154

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*, 2019.

[2] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proc. ICLR*, 2014.

[3] Ziyi Wang, Siting Liu, Yuan Pu, Song Chen, Tsung-Yi Ho, and Bei Yu. Restructure-tolerant timing prediction via multimodal fusion. In *Proc. DAC*, 2023.

[4] Zehao Xiao, Xiantong Zhen, Ling Shao, and Cees GM Snoek. Learning to generalize across domains on single test samples. In *Proc. ICLR*, 2022.

[5] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *Proc. ECCV*, 2014.